Coding Lab 1: Reverse Complement of DNA

Objective

Write a program that takes one or more DNA sequences (in FASTA or plain string form) and returns their reverse complements.

Background

DNA has directionality (5' $\rightarrow$ 3'). The reverse complement of a sequence is obtained by reversing the sequence and replacing each base with its complement (A - T, C - G).

Motivation: Why Compute the Reverse Complement?

DNA is a double-stranded molecule with each strand having a defined directionality:

- One strand runs 5' $\rightarrow$ 3' (often referred to as the **sense** or coding strand).
- The complementary strand runs 3' $\rightarrow$ 5' (called the **antisense** or template strand).

> **Transcription:** RNA polymerase synthesizes mRNA from the **template strand**, which is complementary and antiparallel to the coding strand. If only the sense strand is available, the reverse complement must be computed to simulate transcription.

Understanding and computing the reverse complement is essential in many areas of molecular biology and bioinformatics. The following are key use cases:

- **Primer Design:** In PCR, the reverse primer must bind to the reverse complement of the target region. Designing primers thus requires accurate computation of the reverse complement.
- **Sequence Annotation:** Gene annotations (e.g. in GenBank) may refer to sequences located on the minus strand. To extract the correct nucleotide or amino acid sequence, reverse complementation is necessary.
- **Alignment and Assembly:** Many alignment tools (e.g. BLAST) compare sequences in both forward and reverse orientations. Reverse complements are critical in this process.
- **CRISPR/Cas9 Guide Design:** CRISPR systems rely on base pairing with the target strand. Often, reverse complement logic is used to identify off-target sites and design guide RNAs.
- **Motif Discovery:** When searching for DNA motifs or restriction sites, it is standard practice to examine both the original sequence and its reverse complement.

Tasks

1. Write a function `reverse_complement(seq)` that returns the reverse complement of a DNA string `seq`.
2. Extend it to parse a FASTA file (one or more sequences) and output the reverse-complemented sequences in FASTA format.
3. Add error-checking: reject or warn about invalid characters (e.g. N, lowercase, non-ATCG).

Scaffold Code (Python)

```
def reverse_complement(seq: str) -> str:
    # TODO: implement mapping A <-> T, C <-> G and reverse
    pass

def process_fasta(infile: str, outfile: str):
    # Read FASTA, compute reverse complement, write FASTA.
    with open(infile) as inf, open(outfile, "w") as outf:
        # Simple parser
        header = None
        seq_lines = []
        for line in inf:
            if line.startswith(">"):
                if header:
                    rc = reverse_complement("".join(seq_lines))
                    outf.write(header + "\\n" + rc + "\\n")
                header = line.strip()
                seq_lines = []
            else:
                seq_lines.append(line.strip())
        # last record
        if header:
            rc = reverse_complement("".join(seq_lines))
            outf.write(header + "\\n" + rc + "\\n")

if __name__ == "__main__":
    # Example usage
    seq = "ATCGGTA"
    print(reverse_complement(seq))
```

Hints

- Use a dictionary for complementing,
  - e.g. `complement = 'A':'T','T':'A','C':'G','G':'C'`.
- Reverse by slicing: `seq[::-1]`.
- Be careful with newline characters and whitespace.
- When parsing FASTA, preserve headers and ensure output sequences are in uppercase.

Extensions

- Handle ambiguous bases (e.g. N $\rightarrow$ N).
- Provide both directions (original and reverse complement) in the output file.
- Add command-line interface using `argparse`.